A MOF-based Metamodel and UML Syntax for Networked Ontologies

Saartje Brockmans, Peter Haase, Rudi Studer

Institute AIFB, Universität Karlsruhe (TH), Germany
{brockmans, haase, studer}@aifb.uni-karlsruhe.de

Abstract. Next generation semantic applications will be characterized by a large number of networked ontologies; as a consequence the complexity of semantic applications increases. In this paper we present a metamodel and a UML syntax for the specification of networked ontologies. For the specification of the metamodel, we follow the metamodeling approach of Model Driven Architecures.

1 Introduction

Ontologies provide a key technology to support interoperability on the web and to enable semantic integration of both data and processes. A decade after the notion of ontology was first proposed by Tom Gruber [7], ontologies have come of age, and we are now entering a new phase, one in which ontologies are being produced in larger numbers and exhibit greater complexity than ever before. Next generation semantic applications will be characterized by a large number of ontologies, some of them constantly evolving. As the complexity of semantic applications increases, more and more knowledge will be embedded in applications, typically drawn from a wide variety of sources. This new generation of applications will thus reflect the fact that new ontologies are embedded in a network of already existing ontologies. In contrast with the current model of semantic integration, future applications will rely on networks of contextualized ontologies, which are usually locally, but not globally consistent. Thus, we are now facing both new opportunities and new challenges. Specifically, we now have the opportunity to build systems exhibiting a level of complexity qualitatively superior to the previous generation of semantic systems, by integrating a variety of large, reusable semantic resources. At the same time, we also face a challenge: current methodologies and technologies are simply not sophisticated enough to support the whole application development lifecycle for the envisaged semantic applications. In the current situation, ontologies are distributed over the web, sometimes available directly, sometimes hidden within corporate networks. These ontologies are related to each others, but this remains difficult to assess: some are simple copies of other ones (it is hard to know which one is the master copy), some are versions of others (it is hard to know which one came first), some are used jointly with others (this information is hidden in applications), some are imported by others. A comprehensive understanding of the complex relationships between ontologies in a network is key to be able to model, represent and manage networked ontologies within an ontology infrastructure. Thus, in this paper we provide a definition and formalization of networked ontologies. We focus on three of the main entities that are manipulated in networked ontologies: (1) Ontologies themselves, where we rely on the Web Ontology Language OWL [11], which has by now been well-established in the context of the Semantic Web, (2) Rules, that extend the expressiveness of Description Logics-based ontologies with the capability to model knowledge in the form of horn-like implications, and (3) Mappings, which express relations between ontologies that can be processed by applications (e.g., for importing data or translating messages).

To achieve flexibility and applicability, we define the networked ontology model using a metamodeling approach: The metamodeling features of Model Driven Architecture (MDA, [12]) provide the means for the specification of modeling languages in a standardized, platform independent manner. In short, the Meta Object Facility (MOF, [14]) provides the language for creating metamodels, UML [6] defines the language for creating models corresponding to specific metamodels. Defining the networked ontology model in terms of a MOF compliant metamodel yields a number of advantages:

Interoperability with Software Engineering approaches In order for the networked ontology model to be widely adopted by users and to succeed in real-life applications, it must be well integrated with mainstream software trends. This includes in particular interoperability with existing software tools and applications to put it closer to ordinary developers. MDA is a solid basis for establishing such interoperability. With the networked ontology model defined in MOF, we can utilize MDA's support in modeling tools, model management and interoperability with other MOF-defined metamodels.

Reuse of UML for modeling With respect to interoperability with other metamodels, UML is of particular importance. UML is a well established formalism for visual modeling and recently has been proposed as a visual notation for knowledge representation languages as well [1]. While UML itself lacks specific features of KR languages, the extension mechanisms, UML profiles, allow to tailor the visual notation to the specific required needs.

Independence of particularities of specific formalisms The metamodeling approach of MDA and MOF allows to define the networked ontology model in an abstract form independent of the particularities of specific logical formalisms. This enables to be compatible with currently competing formalisms (e.g. in the case of mapping languages), for which no standard exists yet. Language mappings, also called groundings, define the relationship with particular formalisms and provide the semantics for the networked ontology model. Further, the extensibility capabilities of MOF allow to add new modules to the metamodel if required in the future.

The remainder of the paper is structured as follows: In Section 2 we provide an overview of related work. In Section 3 we introduce the metamodeling features of MOF and how they are used for the specification of the networked ontology model. In Section 4 we present the specification of the networked ontology model, namely the metamodel of OWL DL, which serves as the core of the model, the rules metamodel, which provides support for defining rules on top of ontologies, and the mapping metamodel, which describes how ontologies in a network of ontologies are related via ontology mappings. For a full reference to the complete specification, we refer the reader to [3]. We conclude the paper in Section 5.

2 Related Work

There exists a number of related works for the specification of ontology, rule and mapping languages. However, there currently exist no approaches that allow for the specification of networked ontologies in an *integrated* manner. The most relevant thread of related work are the efforts of the OMG to standardize an Ontology Definition Metamodel (ODM). As a response to the original call of the OMG for an Ontology Definition Metamodel [15], the OMG has received a number of diverse proposals. The various proposals have been merged into one submission [10] that covers several metamodels for RDF, OWL, Common Logic, and Topic Maps, as well as mappings between them. Our proposed metamodel departs from this approach as it strictly focuses on OWL DL and is tailored to its specific features. Further, rule extensions have been considered important for the ODM, but have not been addressed so far. Also, no other proposals so far has considered OWL ontology mappings. Currently, our work is being aligned with the OMG standardization efforts. To the best of our knowledge, our work presents the first MOF-based metamodel and UML profile for rule-extended OWL ontologies together with OWL ontology mappings.

3 Metamodeling with MOF

This section introduces the essential ideas of MOF and shows how a metamodel and a UML profile for networked ontologies fit into this more general picture.

The methodology, tools and technology of MOF and UML seem to be an adequate approach for supporting the development and maintenance of networked ontologies. The general idea of using MOF-based metamodels and UML profiles for this purpose is depicted in Figure 1: A metamodel for networked ontologies as well as a UML profile are grounded in MOF, in that they are defined in terms of the MOF meta-metamodel, explained further in this section. The UML profile mechanism is an extension mechanism to tailor UML to specific application areas. Our proposed UML profile defines a visual notation for optimally supporting the specification of ontologies, rules and ontology mappings. This visual syntax is based on the metamodel and is independent from a concrete mapping formalism. Mappings in both directions between the metamodel and the profile have to be established.

OWL DL ontologies, rules, and ontology mappings in a concrete language instantiate the metamodel. The constructs of the specific languages have a direct correspondence with those of the metamodel. Analogously, specific UML models

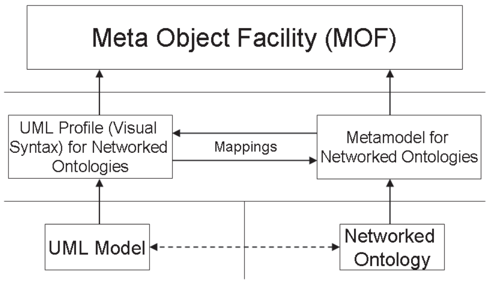
Meta Object Facility (MOF)
UML Profile (Visual Syntax) for Networked Ontologies
Mappings
Metamodel for Networked Ontologies
UML Model
Networked Ontology

Fig. 1. How a metamodel and a UML profile for networked ontologies fit into the picture of the Meta Object Facility framework

instantiate the UML profile. Within the MOF framework, the UML models are translated into definitions based on the above mappings between the metamodel and the UML profile. In case of ontology mappings, the decision about a concrete mapping formalism is taken in this translation step, so after the visual modeling of the ontology mappings. This decision is based on the types of the mappings which were modeled.

3.1 Meta Object Facility

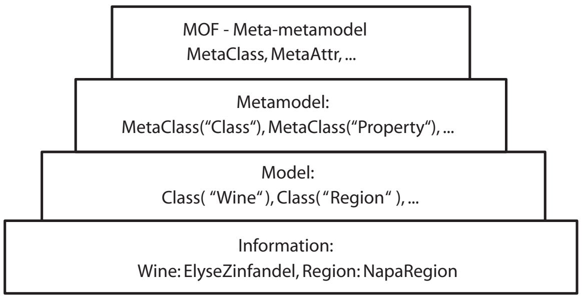
MOF - Meta-metamodel
MetaClass, MetaAttr, ...
Metamodel:
MetaClass("Class"), MetaClass("Property"), ...
Model:
Class("Wine"), Class("Region"), ...
Information:
Wine: ElyseZinfandel, Region: NapaRegion

Fig. 2. OMG Four-Layer Metadata Architecture

The Meta Object Facility (MOF) is an extensible model driven integration framework for defining, manipulating and integrating metadata and data in a platform independent manner. The goal is to provide a framework that supports any kind of metadata and that allows new kinds to be added as required.

MOF plays a crucial role in the four-layer metadata architecture of the Object Management Group (OMG) shown in Figure 2. The bottom layer of this architecture encompasses the raw information to be described. For example, Figure 2 contains information about a wine called ElyseZinfandel and about the Napa region, where this wine grows. The model layer contains the definition of the required structures, e.g. in the example it contains the classes used for grouping information. Consequently, the classes Wine and Region are defined. If these are combined, they describe the model for the given domain. The metamodel defines the terms in which the model is expressed. In our example, we would state that models are expressed with classes and properties by instantiating the respective meta classes. Finally, the MOF constitutes the top layer, also called the meta-metamodel layer. Note that the top MOF layer is hard wired in the sense that it is fixed, while the other layers are flexible and allow to express various metamodels such as the UML metamodel or the metamodel for OWL DL ontologies, rules and ontology mappings.

3.2 Ontology Metamodel and UML Profile

The ontology metamodel as well as a UML profile are grounded in MOF, in that they are defined in terms of the MOF meta-metamodel. The UML profile mechanism is an extension mechanism to tailor UML to specific application areas. UML profiles define a visual notation for optimally supporting the specification of networked ontology models. This visual syntax is based on the metamodel. Mappings in both directions between the metamodel and the profile have to be established.

However, the OWL ontology metamodel is just one part of the networked ontology model. The metamodel consists of several modules. The core module, i.e. the OWL metamodel, is extended by different modules that provide additional features, e.g. rules or mappings. In many application scenarios, only particular aspects of the networked ontology model are needed. In these cases, only the relevant modules need to be supported and used.

While the OWL ontology metamodel has a direct grounding in the OWL ontology language, the extensions have a generic character in that they are formalism independent and allow a grounding in different formalisms.

4 The Networked Ontology Metamodel

In this section, we present the networked ontology metamodel and a corresponding UML profile. We build it incrementally starting with the OWL DL ontology metamodel and UML profile. Secondly we introduce the metamodel and UML profile for SWRL rules, extending these for OWL DL. Finally, we present the metamodel and UML profile for OWL ontology mappings, allowing to model mappings in a formalism-independent way, which is based on the ontology and rules metamodels. In the metamodel figures, darker grey classes denote existing classes from the underlying metamodels. All metamodels are augmented with constraints specifying invariants that have to be fulfilled by all models that instantiate the metamodels. These constraints are expressed in the Object Constraint Language [17], a declarative language that provides constraint and object query expressions on object models that cannot otherwise be expressed by diagrammatic notation. Since the UML profile mechanism supports a restricted form of metamodeling, our proposed UML profiles contain a set of extensions and constraints to UML2. This tailors UML2 such that models instantiating the metamodels can be defined. Extensions to UML2 consist of custom UML stereotypes, which usually carry the name of the corresponding language element, and dependencies. For a complete reference of the formal correspondence between the metamodels and the languages, the relationship between the UML profiles and the metamodels, and the OCL constraints for the metamodels, we refer the reader to [3].

4.1 A Metamodel and UML Profile for Ontologies

A Metamodel for OWL Ontologies Our ontology metamodel [4] defines a metamodel for OWL ontologies. This metamodel is built on the MOF framework. A metamodel for a language that allows the definition of ontologies naturally follows from the modeling primitives offered by the ontology language. The proposed metamodel has a one-to-one mapping to the abstract syntax of OWL DL and thereby to the formal semantics of OWL. It primarily uses basic well-known concepts from UML.

Figure 3 shows the main elements of the ontology metamodel. Every element of an ontology is defined as a subclass of `OntologyElement` and hence as a member of an `Ontology`.

For a full representation of all other elements of the OWL DL metamodel, we refer to [4].

A UML Profile for OWL Ontologies The UML ontology profile describes a visual UML syntax to model ontologies. We provide a UML profile that is faithful to both UML2 and OWL DL, with a maximal reuse of UML2 features and OWL DL features. Our UML profile has a basic mapping, from OWL class to UML class, from OWL property to binary UML association, from OWL individual to UML object, and from OWL property filler to UML object association.

Figure 4 shows a small example of an ontology depicted using the UML profile. It contains the definition of classes `Article`, `Book` and `Thesis` as subclasses of `Publication`. The first two are defined to be disjoint, using an appropriate stereotype on the dependency between both classes. Moreover, the ontology contains a class `Person` and its subclass `Researcher`. An association between `Publication` and `Person` denotes the object property `authorOf`, from which domain and range are defined via an association class. Furthermore, the ontology defines an object property between `Publication` and `Topic`. Finally, the ontology contains some instances of its classes and object properties.

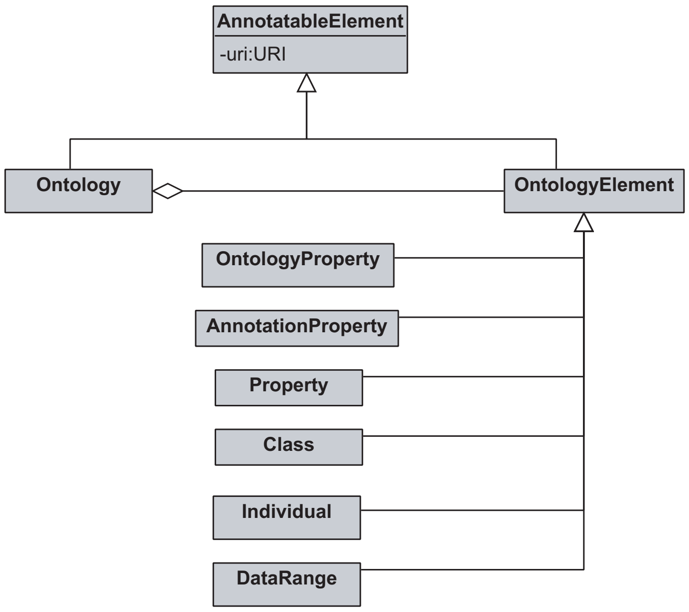

Fig. 3. Main Elements of the Ontology Metamodel

Another small ontology of the same domain is presented in Figure 5. For a discussion of all details of the UML profile for OWL DL ontologies, we refer the reader to [4].

4.2 A Metamodel and UML Profile for Rules

A Metamodel for SWRL Rules We propose a metamodel for rules as a consistent extension of the metamodel for OWL DL ontologies which we described in the previous section. In particular, we focus on SWRL rules, which is currently the most prominent proposal for a rule extension of OWL. It also subsumes other proposals such as DL-safe rules [13]. Figure 6 shows the metamodel for SWRL rules. We discuss the metamodel along the SWRL specifications. Interested readers may refer to the specifications [9] for a full account of SWRL.

SWRL defines rules as part of an ontology. Like that, the SWRL metamodel defines `Rule` as a subclass of `OntologyElement`. `OntologyElement` is defined in the OWL DL metamodel (Figure 3) as an element of an `Ontology`, via the composition link between `NamedElement` and `Ontology`. As can also be seen in Figure 3, the class `OntologyElement` is a subclass of the class `AnnotatableElement`, which defines that rules can be annotated. As annotations are modeled in the ontology metamodel, a URI reference can be assigned to a rule for identification.

Moreover, as Figure 6 shows, a rule consists of an antecedent and a consequent, also referred to as body and head of the rule, respectively. The multiplicities of the association between `Antecedent` and `Atom` and of the association

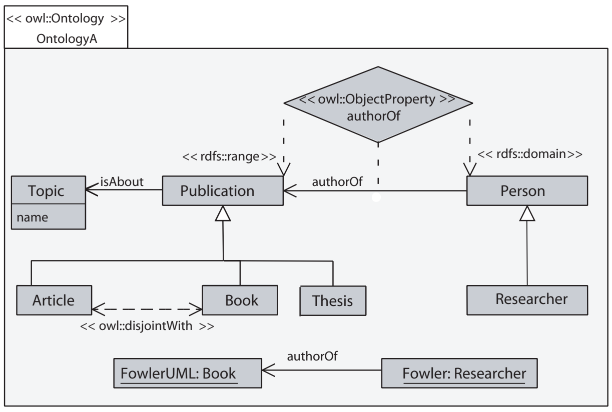

Fig. 4. A first sample ontology depicted using the UML profile for OWL ontologies

between `Consequent` and `Atom` define that an antecedent and a consequent can hold zero or more atoms. The multiplicity in the other direction defines that the same atom can appear in several antecedents or consequents. According to the SWRL specifications, every `Variable` that occurs in the `Consequent` of a rule must occur in the `Antecedent` of that rule, a condition referred to as "safety". We defined this condition in the metamodel as one of the OCL constraints.

The atoms of the antecedent and the consequent consist of predicate symbols and terms. According to SWRL, they can have different forms. The first three, OWL description, data range and property, were already provided in the OWL ontology metamodel, namely as metaclasses `Class`, `DataRange` and `Property`, respectively. As can be seen in Figure 6, the predicates `Class`, `DataRange`, `Property` and `BuiltIn` are all defined as subclasses of the class `PredicateSymbol`, which is associated to `Atom`. The remaining two atom types, `sameAs` and `differentFrom`, are represented as specific instances of `PredicateSymbol`.

To define the order of the atom terms, we put a class `TermList` in between `Atom` and `Term`. This UML association class connects atoms with terms and defines the term order via the attribute `order`.

A UML Profile for Rules In this section, we introduce a UML profile for SWRL rules which is consistent with the design considerations taken for the UML profile for OWL ontologies. Figure 7 shows an example of a rule, which defines that when an author does not like the topic of his publication, he is a bad author.

As can be seen in the Figure, a rule is depicted by two boxes connected via a dependency with the stereotype `rule`. All atoms of the antecedent are contained in the box at the origin of the dependency, whereas the box at the end contains the consequent. This way, antecedent and consequent can easily be

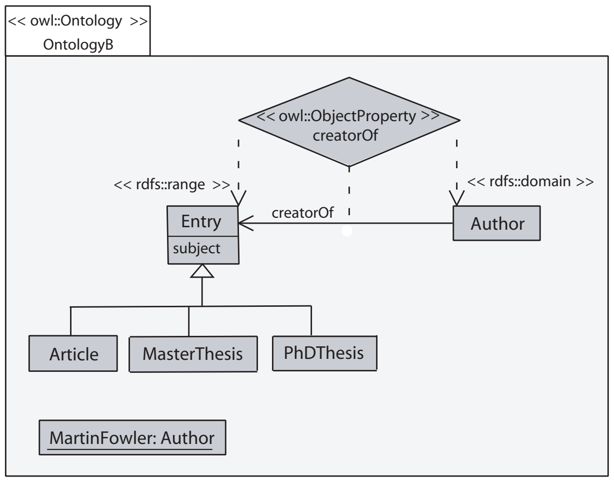

Fig. 5. A second sample ontology depicted using the UML profile for OWL ontologies

distinguished, and it also allows to distinguish between the rule atoms and the OWL DL facts which are depicted in similar ways. The left box of our example contains the three variable definitions and the three properties that are defined between these variables. The consequent-box on the right contains the definition of the variable X from which it is known which class it belongs to.

Although the existing ontology UML profile already comprises a visual syntax for individuals and data values, namely by applying the UML object notation, it does not include a notation for variables since OWL DL ontologies do not contain variables. We decided to depict variables in the UML object notation as well, since a variable can be seen as a partially unknown class instance or data value. We provide a stereotype `variable` to distinguish a variable.

Finally, we now discuss the representation of the different SWRL predicate symbols. For atoms with a class description as its predicate symbol, the UML profile for OWL DL provided a notation for individuals as instances of class descriptions already. An atom with a class description and a variable as its term, is illustrated similarly. An appropriate stereotype is added. An example of this can be seen in the consequent in Figure 7. Similarly, a visual construct for a data range definition using individuals is contained in the UML profile for OWL DL as well, namely represented in the same way as class individuals. Data range constructs containing variables are also depicted in a similar fashion.

Object properties are depicted as directed associations between the two involved elements. A datatype property is pictured as a UML attribute. These notations were provided for properties of individuals by the UML profile for OWL DL, and we follow them to depict properties of variables. The antecedent of the rule in Figure 7 contains three such object properties between variables, `authorOf`, `dislikesTopic` and `isAbout`.

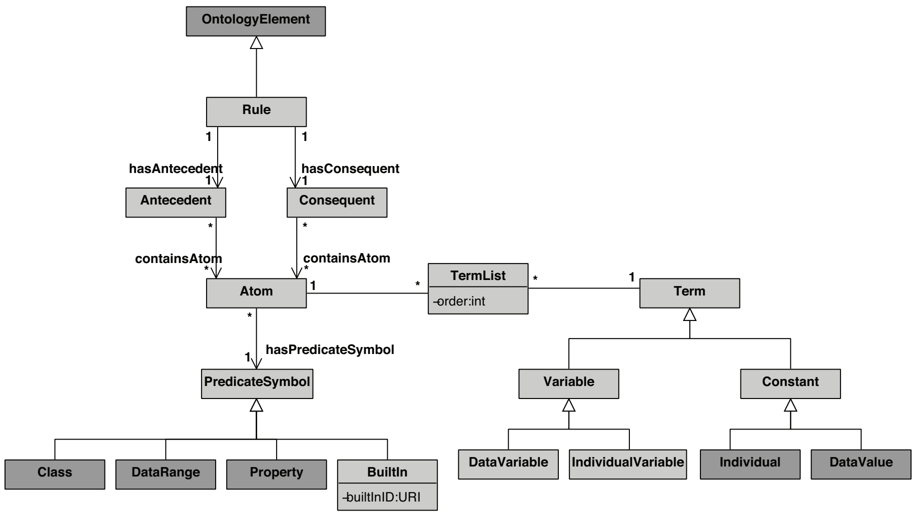

Fig. 6. The Rule Metamodel

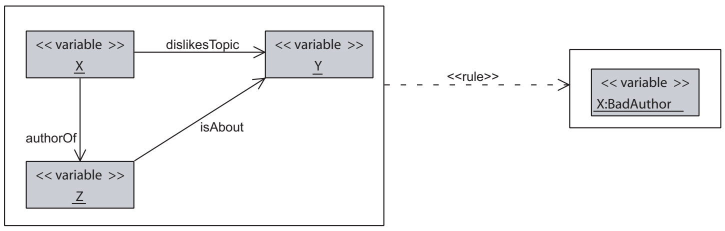

Fig. 7. $\text{BadAuthor}(x) \leftarrow \text{dislikesTopic}(x, y) \wedge \text{authorOf}(x, z) \wedge \text{isAbout}(z, y)$

According to the UML profile for OWL DL, equality and inequality between objects are depicted using object relations. Because of the similarity between individuals and variables, as shortly mentioned before, we propose to use the same visual notation for `sameAs` and `differentFrom` relations between two variables or between a variable and an object.

The last possible predicate symbol in SWRL rules is Built-in. For the visual representation of built-in relations, we use usual associations to all participating variables and data values, similar to the `owl:AllDifferent` concept provided in the UML profile for OWL DL. To denote the built-in relation, we provide the stereotype `built-in` together with the specific built-in ID. The names of the associations denote the order of the arguments, by numbers. For the six most basic built-ins, `swrlb:equal`, `swrlb:notEqual`, `swrlb:lessThan`, `swrlb:lessThanOrEqual`, `swrlb:greaterThan` and `swrlb:greaterThanOrEqual`, we provide an alternative notation. Instead of depicting the stereotype and the name of the built-in, an appropriate icon can be used.

4.3 A Metamodel and UML Profile for Ontology Mappings

A Metamodel for OWL Ontology Mappings In typical use cases such as data translation, data integration, etc. mappings between different rule-extended ontologies would have to be defined. We propose a formalism-independent metamodel for OWL ontology mappings ([2], [8], [5]).The metamodel is a consistent extension of our metamodels for OWL DL ontologies and SWRL rules.

Figure 8 shows the metamodel for mappings. The central class in the meta-

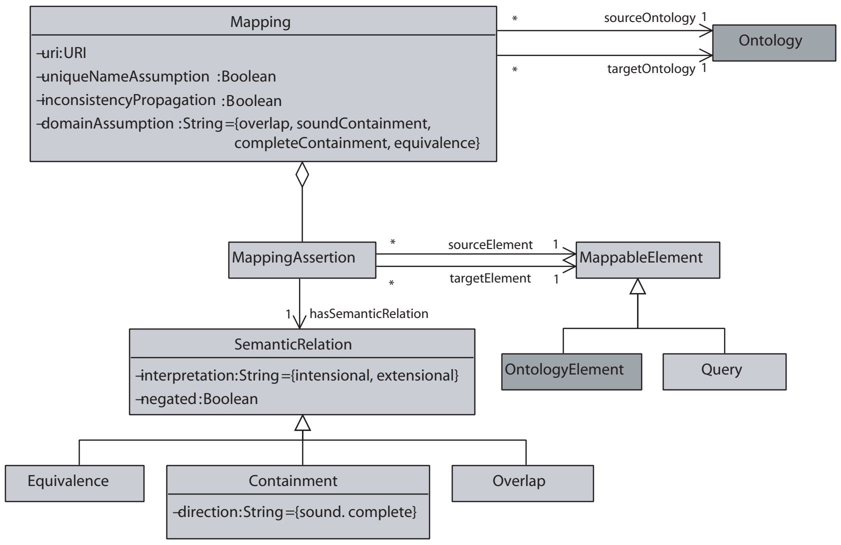
Mapping
-uri:URI
-uniqueNameAssumption :Boolean
-inconsistencyPropagation :Boolean
-domainAssumption :String={overlap, soundContainment, completeContainment, equivalence}
*
sourceOntology 1
*
targetOntology 1
Ontology
MappingAssertion
*
sourceElement 1
*
targetElement 1
MappableElement
1
hasSemanticRelation
SemanticRelation
-interpretation:String={intensional, extensional}
-negated:Boolean
OntologyElement
Query
Equivalence
Containment
-direction:String={sound. complete}
Overlap

Fig. 8. The Ontology Mapping Metamodel

model is the class `Mapping`, having four attributes, of which `uri` allows to uniquely identify a mapping and refer to it as a first-class object.

A mapping is always defined between two ontologies. An ontology is represented by the class `Ontology` in the OWL DL metamodel. Two associations from `Mapping` to `Ontology`, `sourceOntology` and `targetOntology`, specify the source respectively the target ontology of the mapping. Cardinalities on both associations denote that to each `Mapping` instantiation, there is exactly one `Ontology` connected as source and one as target.

A mapping consists of a set of mapping assertions, denoted by the MOF aggregation relationship between the two classes `Mapping` and `MappingAssertion`. The elements that are mapped in a `MappingAssertion` are defined by the class `MappableElement`. A `MappingAssertion` is defined through exactly one `SemanticRelation`, one source `MappableElement` and one target `MappableElement`. This is defined through the three associations starting from `MappingAssertion`, and their cardinalities.

We define four semantic relations along with their logical negation to be defined in the metamodel. Equivalence and Overlap are directly contained in the metamodel as subclassesof the class SemanticRelation. The other two, containment in either direction, are defined through the subclass Containment and its additional attribute direction, which can be sound or complete.

The negated versions of all semantic relations are specified through the boolean attribute negated of the class SemanticRelation. For example, a negated Overlap relation specifies the disjointness of two elements. The other attribute of SemanticRelation, interpretation, defines whether the mapping assertion is assumed to be interpreted intentionally or extensionally. Please note that the metamodel in principle supports all semantic relations for all mappable elements, including individuals.

A mapping assertion can connect two mappable elements, which may be ontology elements or queries. To support this, MappableElement has two subclasses OntologyElement and Query. The former is previously defined in the OWL DL metamodel, as shown in Figure 3. The class Query reuses constructs from the SWRL rules metamodel (Section 4.2). The reason for reusing large parts of the rule metamodel lies in the fact that conceptually, rules and queries are of very similar nature [16]: A rule consists of a rule body (antecedent) and rule head (consequent), both of which are conjunctions of logical atoms. A query can be considered as a special kind of rule with an empty head. The distinguished variables specify the variables that are returned by the query. Informally, the answer to a query consists of all variable bindings for which the grounded rule body is logically implied by the ontology.

A UML Profile for OWL Ontology Mappings This section describes the UML profile for specifying ontology mappings. Our goal is to allow the user to specify mappings without having decided yet on a specific mapping language or even on a specific semantic relation. This is reflected in the proposed visual syntax which is, like the metamodel, independent from a concrete mapping formalism. After specifying the mappings, the decision on the specific mapping formalism is being taken. The UML profile is consistent with the design considerations taken for the previously defined UML profiles for OWL ontologies and SWRL rules.

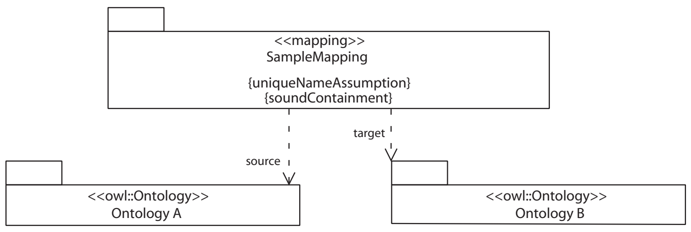

Fig. 9. Visual notation for a mapping between two ontologies

First of all, users specify two ontologies between which they want to define mappings. The visual notation for this as defined in our profile, is presented in Figure 9. Just as for ontologies as collections of ontology elements, we apply the UML grouping construct of a package to represent mappings as collections of mapping assertions. Attributes of the mapping, like the domain assumption, are represented within curly brackets.

In Figure 10, a source concept `Publication` is defined to be more specific than the target concept `Entry`.

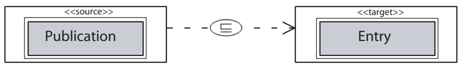

Fig. 10. Sample containment relation between two concepts

Both source and target elements of mapping assertions are represented in a box, connected to each other via a dependency with the corresponding symbol of the semantic relation. In the first step of the process, when users just mark elements being semantically related without specifying the type of semantic relation, the dependency does not carry any relation symbol. Stereotypes in the two boxes denote source- and target ontology. Like defined in the metamodel, these mapped elements can be any element of an ontology (metaclass `OntologyElement`) or a query (metaclass `Query`). They are represented like defined in the UML profile for OWL and rules. The parts of the mappable elements which are effectively being mapped to each other, are denoted via a double-lined box, which becomes relevant if the mapped elements are more complex constructs, as explained in the following.

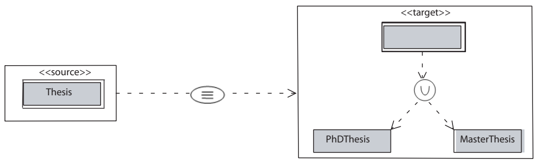

Fig. 11. Sample equivalence relation between complex class descriptions

A more complex example mapping assertion is pictured in Figure 11. The example defines that the union of the classes `PhDThesis` and `MasterThesis`, is equivalent to the class `Thesis`. Figure 12 shows an example of an equivalence relation between two queries. The first query is about a Publication X with a Topic Y named Z. The target query is about an Entry X with subject Z. The mapping assertion defines the two queries to be equivalent. The effective correspondences are established between the the two distinguished variables X and Z, which are again denoted with a double-lined box.

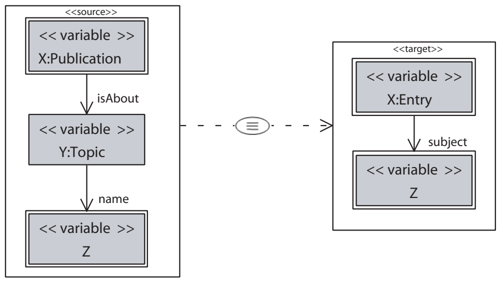

Fig. 12. Sample equivalence relation between two queries

5 Conclusion

Next generation semantic applications will be characterized by a large number of networked ontologies; as a consequence the complexity of semantic applications increases. In this paper we have presented a metamodel for the specification of networked ontologies. We have followed the metamodeling approach of Model Driven Architecures for the specification of the metamodel. Specifically, we have presented three modules of the metamodel:

1. The OWL metamodel serves as the core of our networked ontology model,
2. the rule metamodel extends the ontology language with the expressiveness to model horn-like rules, and
3. the mapping metamodel allows to specify ontology mappings that describe correspondences between ontology elements in a network of ontologies.

The specification of the metamodel only is a first step. Future Work will include:

– Extensions and refinements of the current metamodel to deal with additional aspects of networked ontologies, such as an explicit representation of context.
– The alignment of our metamodel with standardization activities of the OMG. As mentioned before, first steps in this direction have been taken.
– The alignment with future results of the W3C Rule Interchange Format (RIF) Working Group[1]. Currently, we support SWRL as one of the RIF proposals.
– The implementation of components to support the individual phases of the lifecycle of networked ontologies, including modeling, maintaining, and evolving networked ontologies.

Acknowledgments

The work reported in this paper was partially funded by the European Union in the IST projects NeOn (IST-2006-027595, `http://www.neon-project.org/`) and Sekt (IST-2003-506826, `http://www.sekt-project.com/`) and the German Research Foundation (DFG) in scope of Graduate School Information Management and Market Engineering.

[1] `http://www.w3.org/2005/rules/`

References

1. K. Baclawski, M. Kokar, Kogut P., Hart J., Smith J., Holmes W., J. Letkowski, and Aronson M. Extending UML to Support Ontological Engineering for the Semantic Web. In *4th International Conference on UML*, Toronto, Canada, October 2001.
2. P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini, and H. Stuckenschmidt. C-OWL: Contextualizing ontologies. In *Second International Semantic Web Conference ISWC'03*, volume 2870 of *LNCS*, pages 164–179. Springer, 2003.
3. S. Brockmans and P. Haase. A Metamodel and UML Profile for Networked Ontologies – A Complete Reference. Technical report, Universität Karlsruhe, April 2006. `http://www.aifb.uni-karlsruhe.de/WBS/sbr/publications/ontology-metamodeling.pdf`.
4. S. Brockmans, R. Volz, A. Eberhart, and P. Loeffler. Visual Modeling of OWL DL Ontologies using UML. In F. van Harmelen, S. A. McIlraith, and D. Plexousakis, editors, *The Semantic Web – ISWC 2004*, pages 198–213. Springer-Verlag, 2004.
5. D. Calvanese, G. De Giacomo, and M. Lenzerini. Description logics for information integration. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2408 of *LNCS*, pages 41–60. Springer, 2002.
6. Martin Fowler. *UML Distilled*. Addison-Wesley, third edition, 2004.
7. T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.
8. Peter Haase and Boris Motik. A mapping system for the integration of OWL-DL ontologies. In *In Proceedings of the ACM-Workshop: Interoperability of Heterogeneous Information Systems (IHIS05)*, November 2005.
9. Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML*. World Wide Web Consortium, May 2004. W3C Member Submission, `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`.
10. IBM, Sandpiper Software Inc. *Ontology Definition Metamodel, Sixth Revised Submission to OMG*, June 2006.
11. D. L. McGuinness and F. van Harmelen. OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium (W3C), August 2003. Internet: http://www.w3.org/TR/owl-features/.
12. Stephen J. Mellor, Scott Kendall, Axel Uhl, and Dirk Weise. *MDA Distilled*. Addison-Wesley Pub Co, March 2004.
13. B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. In S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *International Semantic Web Conference*, volume 3298 of *Lecture Notes in Computer Science*, pages 549–563. Springer, 2004.
14. Object Management Group. Meta Object Facility (MOF) Specification. Technical report, Object Management Group (OMG), April 2002. `http://www.omg.org/docs/formal/02-04-03.pdf`.
15. Object Management Group. Ontology definition metamodel – request for proposal, March 2003. `http://www.omg.org/docs/ontology/03-03-01.rtf`.
16. S. Tessaris and E. Franconi. Rules and queries with ontologies: a unifying logical framework. In I. Horrocks, U. Sattler, and F. Wolter, editors, *Description Logics*, volume 147 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2005.
17. J. Warmer and A. Kleppe. *Object Constraint Language 2.0*. MITP Verlag, 2004.